Communications of the Association for Information Systems

Volume 48 Article 6

2-2-2021

Programming in a Pandemic: Attaining Academic Integrity in Online Coding Courses

David Goldberg
San Diego State University, dgoldberg@sdsu.edu

Recommended Citation

Goldberg, D. (2021). Programming in a Pandemic: Attaining Academic Integrity in Online Coding Courses. Communications of the Association for Information Systems, 48, pp-pp. https://doi.org/10.17705/1CAIS.04807

Communications of the Association for Information Systems

Research Article DOI: 10.17705/1CAIS.04807 ISSN: 1529-3181

Programming in a Pandemic: Attaining Academic Integrity in Online Coding Courses

David M. Goldberg
Department of Management Information Systems
Fowler College of Business
San Diego State University
dgoldberg@sdsu.edu

Abstract:

The coronavirus disease of 2019 (COVID-19) pandemic has necessitated a transition to online courses, prompting widespread consequences for higher education. Ensuring academic integrity poses a serious concern in these circumstances. Drawn from my experiences teaching online programming courses, I discuss the considerable and manifold flaws in our current anti-cheating measures. I propose a series of strategies that instructors can pursue to make assessments more resilient to cheating. Although there is no panacea, we must begin by acknowledging the problem facing us and discussing earnestly how we can refortify academic integrity.

Keywords: Academic Integrity, Cheating, Online Courses, Virtual Instruction.

This manuscript underwent peer review. It was received 6/24/2020 and was with the authors for one month for one revision. Craig Van Slyke served as Associate Editor.

1 Introduction

In early 2020, the coronavirus disease of 2019 (COVID-19) pandemic forced universities across the world to transition courses to an online format. Enacting this transition with so little warning proved a herculean task, and reimagining coursework planned around face-to-face interactions required substantial retooling. The future is unfortunately subject to many unknowns; my university is one of many embracing a fully online fall semester, and some others are planning hybrid formats. Despite the unexpected circumstances, change has long been coming; online instruction has gained traction in recent years as a means to offer larger courses and meet demand from students across the world (Ortagus & Yang, 2018).

I teach Python-based programming courses with 40 to 60 students at both the 300 (third-year undergraduate) and 500 (master's) levels. These courses began in traditional face-to-face format in spring 2020 before transitioning to a synchronous online format. Although I structured my lectures similarly, I greatly feared how I would ensure academic integrity in the transition. Other instructors in my academic circles commonly fear more frequent cheating in online courses. Instructors across the world have suggested various measures to help, most notably technological anti-cheating solutions.

Despite our best intentions, I find our measures thus far wholly insufficient. I contend that cheating in online courses, and particularly in online programming courses, is more widespread than most realize. I argue that the anti-cheating measures many of us rely upon are at best slight deterrents, and often they are utterly toothless. Recognizing this problem, we cannot simply look the other way and hope that most students choose not to cheat. Doing so would be a disservice to students that play by the rules, and we may be disappointed with the proportion of students whose cost-benefit analyses are decided in favor of the less ethical option.

Guided by my experiences teaching programming courses online, I offer a series of suggestions on how these courses can be structured to protect academic integrity. I hope that these thoughts will help others examine and carefully craft assessment strategies. I hope too that many of my suggestions transcend programming and may be of use to those in other disciplines.

2 The Inconclusive State of Cheating in Online Courses

While students have long cheated on assignments, detecting and preventing cheating has become particularly difficult in the era of online courses. It is of course easier to be confident that a student's work is their own if we have physically watched them complete it. Students know this, overwhelmingly opining that it is easier to cheat in online courses than in face-to-face courses (Harton, Aladia, & Gordon, 2019). However, assessments that have examined the difference in scores between the two formats tend to be noisy. Cavanaugh and Jacquemin (2015) found that online students score on average 0.39 GPA points better, but they attributed some of this difference to other factors, such as the students' backgrounds. Fask, Englander, and Wang (2014) studied the difference in scores on the exam level and found that students scored over 10 percent higher on online exams than face-to-face equivalents. Yet, others such as Stack (2015) have found no significant difference in scores between online and face-to-face exams. Apart from Cavanaugh and Jacquemin (2015), these studies have typically employed small sample sizes from a single instructor and discipline, which may account for the variability in findings. Unfortunately, even a clearer consensus on the difference in scores between online and face-to-face formats would not necessarily quantify the extent of cheating. For instance, a lurking factor is that the quality of instruction may differ in online settings (Fask et al., 2014).

Some have suggested surveying students on their willingness to cheat in online courses, but students' fear that the question is a trap (Bucciol, Cicognani, & Montinari, 2020) and social desirability bias (Larson, 2019) likely lead to underreporting. Likewise, evaluating the efficacy of anti-cheating strategies poses measurement challenges as well. For instance, Mason, Gavrilovska, and Joyner (2019) suggested discussing honor code expectations with students in hopes of reducing cheating. They found that explicitly discussing the honor code in computer science courses resulted in fewer instances of cheating flagged by MOSS, an automated code comparison engine. Yet, they conceded that students may have simply cheated more carefully knowing that their instructors were especially vigilant. Students may have colluded, then modified their code just enough to avoid detection.

3 The Online Cheating Marketplace

Some cheating strategies are not especially novel, such as asking a friend for their homework or sneaking notes under a desk during an exam. More recently, though, a marketplace for cheating has arisen online, and the answers to many questions sit at students' fingertips. Websites such as Chegg, Course Hero, and Quizlet offer students a forum to view solutions to problems and upload their own new problems and solutions. Although many such platforms market themselves as "study tools", they contain the answer keys for specific exams offered by specific instructors (Golden & Kohlbeck, 2020). Textbook-derived questions and solutions are especially nefarious as students can easily obtain and share them online.

For programming courses specifically, the "work-for-hire" strategy in which students pay others to write their code represents an enormous concern (Boese, 2016). Online marketplaces such as Fiverr and Freelancer host vendors openly advertising that they will complete students' assignments for them for a fee. I learned that one student had used work-for-hire to complete a homework assignment this past semester when I searched a phrase from the instructions online. I quickly found that that the student hoped to hire someone not only to complete this specific homework assignment but also the remainder of the homework in my course (see Figure 1).

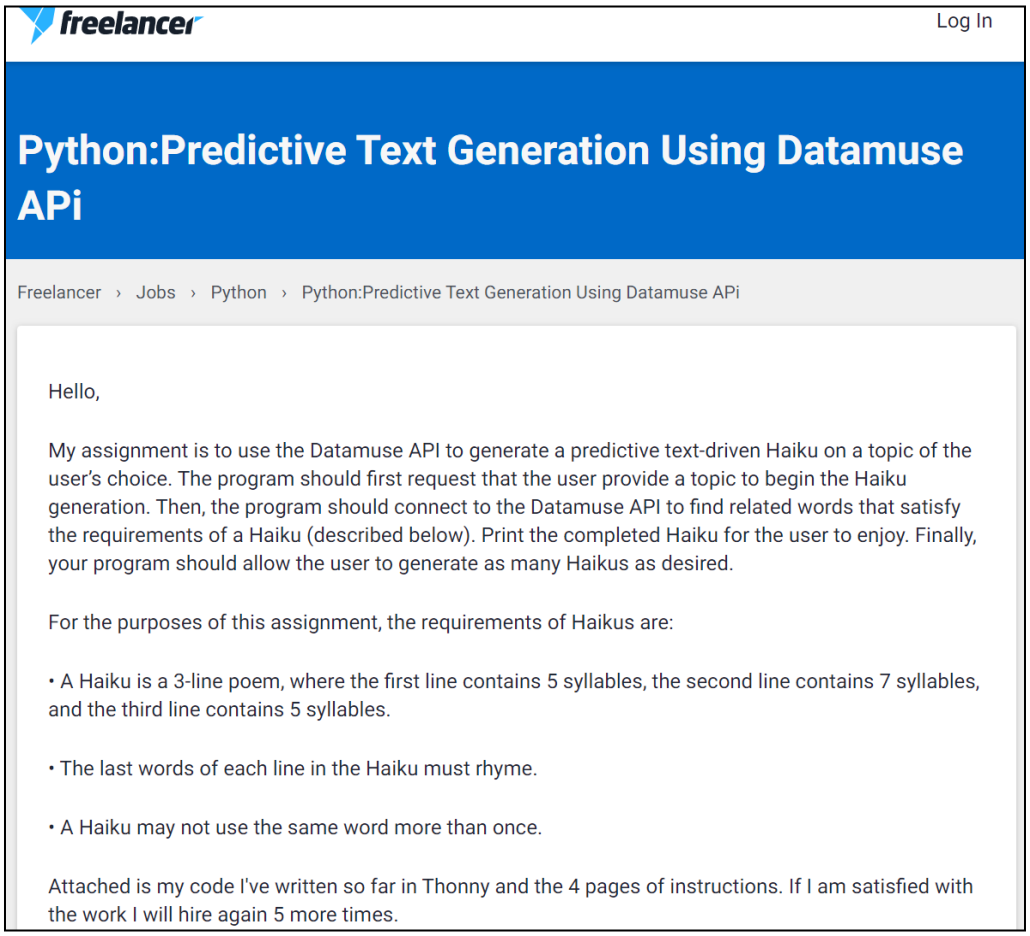

Figure 1. A Post Soliciting Homework Solutions

After the initial pang of existential dread at seeing my assignment online, I worried over possible resolutions. I ultimately decided to confront the class and inform them of my finding. I stated that work-for-hire was unequivocally unacceptable, reiterated the honor code and my expectations, and believed from my students' responses that I had their backing. Yet, as I did so, I had another worry that I have yet to reconcile: had I just taught my class to cheat more covertly? That is, would the work-for-hire scheme end, or would the perpetrator simply find an alternative platform to cheat less publicly? I caught no further work-for-hire schemes over the remainder of the semester, but I can only hope that they were not continued in the shadows.

4 The Latest Anti-cheating Measures

Many instructors have recently turned to technological solutions to alleviate cheating. Respondus Lockdown Browser is an oft-suggested technology that prevents students from accessing any computer resources other than the assessment being completed; of course, this does not prevent them from using a second computer, tablet, phone, or even another person in the room. Recognizing this, others have proposed supplemental video proctoring technologies. One such option, Respondus Monitor, records students through their webcams and intelligently analyzes footage for signs of cheating. ProctorU, a leading alternative, requires students to pay for human proctors to watch them complete assessments through their webcams.

These solutions, though, are problematic both in principle and in practice. Security-minded faculty are quick to remind students to be careful of what they install on their computers, and rightly so. It follows then that anti-cheating software with administrative privileges raises inherent security concerns, and an instructor-mandated webcam peering into a student's home may compromise their privacy. Several students have expressed this reservation to me; am I to reply that they should follow the security practices we preach only when we find it convenient?

My experience confirms that the application of anti-cheating software is just as rife with concerns. Several of my students have struggled with connection issues when completing video-proctored exams. Perhaps unsurprisingly, students with unstable home Internet connections may lack the technological resources to upload live video while completing an exam. I find it unconscionable to place the burden of resolving these problems on students. If students did not volunteer for the online modality, then we cannot reasonably expect them to furnish their homes with sufficient technology. Amid a pandemic with far-ranging consequences on both economics and mobility, it is impossible for many students to purchase better equipment or use library resources. Furthermore, these technological issues are not random; they are particularly concentrated among low-income, minority, and rural students (Horrigan, 2014).

Moreover, exploits exist for even the highest-quality anti-cheating software. One need look no further than Reddit (see Figure 2) to see the technical prowess that our students have developed in circumventing anti-cheating software. Teaching technical skills is a double-edged sword, apparently.

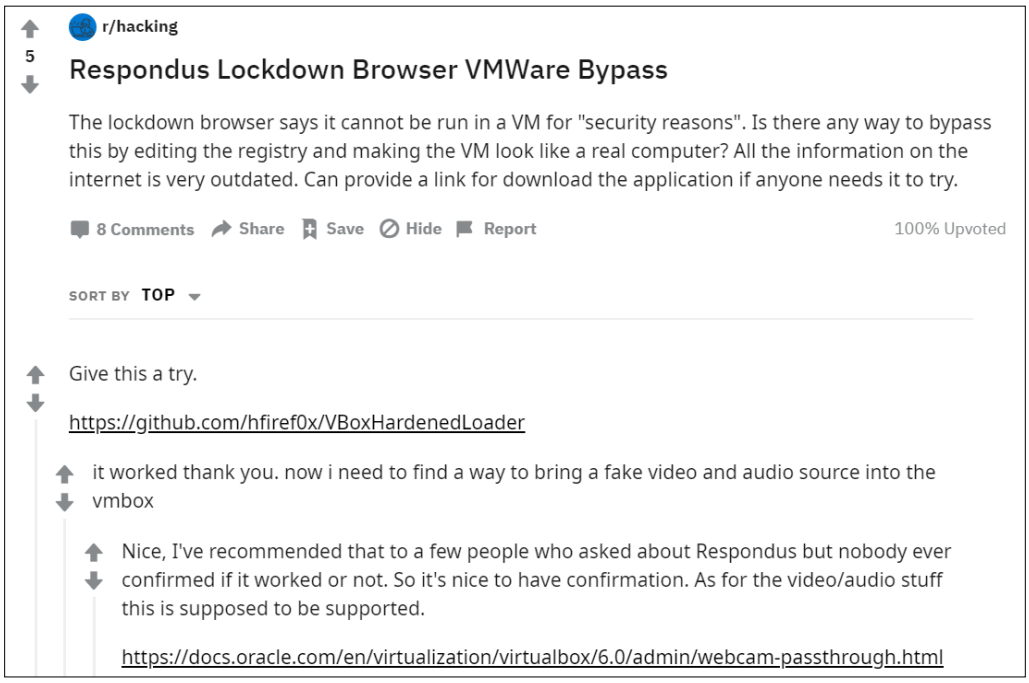

Figure 2. A Thread on Bypassing Anti-cheating Technologies

Alessio and Maurer (2018) have performed the most comprehensive analysis yet of video proctoring's efficacy in online courses, examining its impacts on grade distributions across courses, instructors, and departments. The authors directly compared video-proctored and unproctored versions of each instructor's online course. If video proctoring had substantial efficacy, we would expect video-proctored

sections to score lower both on average and in direct comparisons between video-proctored and unproctored sections of the same course. The authors found lower scores in video-proctored sections on average, but only by 2.2 percent. When they compared course by course, they found that video-proctored sections had significantly lower scores than corresponding unproctored sections 41 percent of the time, that the scores did not significantly differ 28 percent of the time, and somehow that video-proctored sections had significantly *higher* scores than corresponding unproctored sections 28 percent of the time. The efficacy of video proctoring appears marginal at best.

5 Lessons learned

I do not purport to have all the answers, and I would view anyone who claimed that they did with suspicion. Nonetheless, there are some simple adjustments that we can practice in pursuit of academic integrity. On quizzes and exams, I propose the following:

- **Abandon video proctoring**: the technologies have too many problems, and there is scant evidence that they prevent cheating. Lockdown software raises fewer concerns and at least inconveniences cheaters.
- **Eliminate exams in favor of smaller and more frequent quizzes**: cheating behavior is likelier in higher-stakes environments (Kajackaite & Gneezy, 2017). Students who risk earning a low score can more easily justify their cheating on an exam worth 50 percent of their grade than on a quiz worth five percent of their grade.
- **Avoid giving students too much time**: searching for an answer online or communicating an answer with a friend requires time. Instructors can reduce the amount of time students have to cheat by ensuring that they make an assessment available online only for a narrow window and by compressing the time allotted to complete it.
- **Use randomized question pools**: while some overlap between students' assessments will inevitably occur, using learning management systems' randomization capabilities to reduce overlap makes cheating less practical. However, instructors must ensure that they design assessments to guarantee similar difficulty levels for all students.
- **Never use textbook-derived questions**: these questions can easily allow students to obtain an answer key before they perform an assessment.
- **Reduce or eliminate multiple choice questions**: while students can easily collude on multiple choice questions, which have a single "right answer", instructors cannot easily catch such collusion (Manoharan, 2019). Instead, instructors should replace these questions with open-ended prompts that require students to demonstrate their knowledge. In my programming courses, I do so in three ways. First, I ask students to explain and apply a concept from class; for example, "describe stemming and lemmatization, and explain how you would evaluate which to use in a real-world problem". While this question would be straightforward for a student who knew the content, it has enough nuance to pose difficulties if a student tried to quickly search online or communicate with a friend for the answer. Second, I provide students a code snippet and ask them to determine its output. The answers remain open-ended, and students must demonstrate an important ability to understand others' code. Third, I ask students to write some code in text boxes to solve small problems. I am honest with students that I will be lenient on small syntactical issues as I realize they cannot debug; I want to see that they understand key ideas and incorporate them into logical code. Despite some initial trepidation, almost all my students found that they did not find it as daunting to write five to 10 lines of code on quizzes as they feared.

Likewise, on homework and projects, I propose the following:

- **Create real-world requirements**: asking students to, for example, recursively generate values from the Fibonacci sequence may make for a good in-class example, but they can readily find solutions to that problem online. Instead, I ask students to demonstrate their skills in a real-world problem. For instance, this past semester, I asked my students to simulate the financial status of a small farm. The farm grew several crops whose success depended on (pseudo)random chance, and the simulation even incorporated crop rotations. The simulation would end when the farm made enough money for the farmer to "cash out". Students enjoyed

the practical exercise in which they had to think through several layers of logic and structure. Thoroughly addressing detailed business requirements provides valuable experience, and the problem's originality meant that students could not easily obtain the answer to it online.

- **Search assignment instructions online**: I suspect that students can find many other instructors' assignments on websites such as on Freelancer and Chegg. If vigilant, one can have these assignments swiftly removed on copyright grounds. I worry that students may still cheat covertly, but we must prevent what we can.
- **Require students to show their work**: in industry, programmers use version control extensively to track their changes over time. By having students do the same, we both teach them a valuable real-world skill and see their thought process as they complete assignments. With diminished ability to monitor students in online courses, this approach supplements students' scores by recording their progress. A student that truly completed the assignment will make commits in parts, iteratively testing and building piece by piece.
- **Use automated plagiarism checkers**: in programming courses, packages such as MOSS or pycode-similar, while imperfect, can flag the most similar submissions. In non-programming courses, technologies such as Turnitin or Viper can serve the same role. Others have raised concerns over surrendering students' intellectual property to Turnitin (Morris & Stommel, 2017); the decision over an appropriate plagiarism checker ought to be deliberate and consider the course's context.

In closing, I acknowledge that my recommendations place an additional burden on adopters to rethink and redesign their assessments. Navigating these concerns is nontrivial but vital. While we may each play only an isolated role in each student's education, we all contribute to our institutions' academic cultures. It weakens the dutiful work of those who foster academic integrity if others leave their courses vulnerable to cheating and look the other way. Promoting academic integrity may be thankless, but we do no favors for honest students if we equate their work and the work of cheating classmates. We owe it to our students, our institutions, and ourselves to do everything we can to get this right.

References

Alessio, H., & Maurer, K. (2018). The impact of video proctoring in online courses. *Journal on Excellence in College Teaching*, *29*(4), 183-192.

Boese, E. (2016). Just-in-time learning for the just Google it era. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education.*

Bucciol, A., Cicognani, S., & Montinari, N. (2020). Cheating in university exams: The relevance of social factors. *International Review of Economics*, *67*, 319-338.

Cavanaugh, J. K., & Jacquemin, S. J. (2015). A large sample comparison of grade based student learning outcomes in online vs. face-to-face courses. *Online Learning*, *19*(2), 25-32.

Fask, A., Englander, F., & Wang, Z. (2014). Do online exams facilitate cheating? An experiment designed to separate possible cheating from the effect of the online test taking environment. *Journal of Academic Ethics*, *12*(2), 101-112.

Golden, J., & Kohlbeck, M. (2020). Addressing cheating when using test bank questions in online classes. *Journal of Accounting Education*, *52*.

Harton, H. C., Aladia, S., & Gordon, A. (2019). Faculty and student perceptions of cheating in online vs. traditional classes. *Online Journal of Distance Learning Administration*, *22*(4).

Horrigan, J. (2014). Schools and broadband speeds: An analysis of gaps in access to high-speed Internet for African American, Latino, low-income, and rural students. *Alliance for Excellent Education*. Retrieved from https://ecfsapi.fcc.gov/file/60000979777.pdf

Kajackaite, A., & Gneezy, U. (2017). Incentives and cheating. *Games and Economic Behavior*, *102*, 433-444.

Larson, R. B. (2019). Controlling social desirability bias. *International Journal of Market Research*, *61*(5), 534-547.

Manoharan, S. (2019). Cheat-resistant multiple-choice examinations using personalization. *Computers & Education*, *130*, 139-151.

Mason, T., Gavrilovska, A., & Joyner, D. A. (2019). Collaboration versus cheating: Reducing code plagiarism in an online MS computer science program. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education.*

Morris, S. M., & Stommel, J. (2017). A guide for resisting EdTech: The case against Turnitin. *Hybrid Pedagogy*. Retrieved from https://hybridpedagogy.org/resisting-edtech/

Ortagus, J. C., & Yang, L. (2018). An examination of the influence of decreases in state appropriations on online enrollment at public universities. *Research in Higher Education*, *59*(7), 847-865.

Stack, S. (2015). Learning outcomes in an online vs. traditional course. *International Journal for the Scholarship of Teaching and Learning*, *9*(1).

About the Authors

David M. Goldberg is Assistant Professor of Management Information Systems in the Fowler College of Business at San Diego State University. He received his doctoral and bachelor's degrees from Virginia Tech. His current research interests are in the areas of text mining, machine learning, decision support systems, and expert systems. He has published in *Decision Support Systems*, *Decision Sciences, Information Technology & Management*, *Safety Science*, *International Journal of Information Technology & Decision Making*, and others.